gV

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/724,254	12/01/2003	Georg Biehler	Q78306	1606

23373 7590 02/08/2007
SUGHRUE MION, PLLC
2100 PENNSYLVANIA AVENUE, N.W.
SUITE 800
WASHINGTON, DC 20037

EXAMINER	
WANG, BEN C	
ART UNIT	**PAPER NUMBER**
2192	

SHORTENED STATUTORY PERIOD OF RESPONSE	MAIL DATE	DELIVERY MODE
3 MONTHS	02/08/2007	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

If NO period for reply is specified above, the maximum statutory period will apply and will expire 6 MONTHS from the mailing date of this communication.

Office Action Summary	Application No.	Applicant(s)	
	10/724,254	BIEHLER ET AL.	
	Examiner	Art Unit	
	Ben C. Wang	2192	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *01 December 2003*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-44* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-44* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)

2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)

3) ☒ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date *12/01/2003*.

4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .

5) ☐ Notice of Informal Patent Application

6) ☐ Other: _____.

DETAILED ACTION

1. Claims 1-44 are pending in this application and presented for examination.

Claim Rejections – 35 USC § 103(a)

2. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which the subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made

3. Claims 1-44 are rejected under 35 U.S.C. 103(a) as being unpatentable over Logan, III et al., (hereinafter 'Logan') (Pub. No. US 2004/0205702 A1) in view of K. J. Hines (hereinafter 'Hines') (Pub No. US 2005/0246682 A1).

4. **As to claim 1**, Logan discloses a programming tool (Fig. 5; [0001]; [0005]; [0038], Lines 1-3) for at least one of creating (Fig. 5, element 142) and displaying programs (Fig. 5, elements 142, 136; [0005]; [0039], Lines 3-5) to control a flow of a process ([0038], Lines 13-17; [0041], Lines 1-3) using a graphics language for simultaneous representation in a diagram (Fig. 3, element 34; Figs. 7-8), on a display device (Fig. 5, element 136).

But Logan does not disclose a sequence over time and interactions of objects that are involved in the control of the process, wherein a coordination element is provided, which manages the sequence over time and the interactions of the objects involved.

However, in an analogous art, Hines discloses a sequence over time (Fig. 25, elements 2502, 2504; [0401], Lines 1-4) and interactions of objects that are involved in the control of the process (Fig. 25, element 2512; [0401], Lines 16-18), wherein a coordination element is provided ([0125], Lines 1-2; [0135], Lines 1-8; [0176], Lines 1-2, 5-13; [0221], Lines 1-6), which manages the sequence over time (Fig. 25, elements 2502, 2504; [0401], Lines 1-4) and the interactions of the objects involved (Fig. 25, element 2512; [0401], Lines 16-18).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Logan and the teachings of Hines to further provide a sequence over time and interactions of objects that are involved in the control of the process, wherein a coordination element is provided, which manages the sequence over time and the interactions of the objects involved in Logan system.

The motivation is the need remains for a design and programming methodology that allows for coordination between software elements without tailoring the software element to the specific coordination style used in a particular software system while allowing for interactions between software elements is a way that facilitates debugging complex systems as once suggested by Hines ([0030]).

5. **As to claim 19**, Logan discloses a method for programming (Fig. 4A, element 104; [0036], Lines 1-8) and representing a program (Fig. 4A, element 100) run for at least one of open-loop and closed-loop control of a process (Fig. 4A, elements 100, 109; [0036], Lines 8-15), using at least one programmable controller (Fig. 3, elements 90, 98; [0027]; Fig. 5, element 144; [0039]), in which a graphics language (Fig. 3, element 94) is used to implement a process capable of being represented by objects and object interactions (Figs. 7-8), comprising: calling a plurality of objects involved in the process in a common diagram; calling a plurality of respectively required object interactions in the common diagram; editing the selected objects and object interactions (Fig. 5, element 142; [0033], Lines 7-17); and translating the previously implemented program into at least one of a corresponding high-level language [0036], Lines 1-8; [0043], Lines 1-5) and a corresponding machine language ([0019], Lines 1-7).

But Logan does not disclose the sequence of the object interactions over time in the common diagram.

However, in an analogous art, Hines discloses the sequence of the object interactions over time in the common diagram (Fig. 25, elements 2502, 2504; [0401], Lines 1-4).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Logan and the teachings of Hines to further provide the sequence of the object interactions over time in the common diagram in Logan system.

The motivation is the need remains for a design and programming methodology that allows for coordination between software elements without tailoring the software element to the specific coordination style used in a particular software system while allowing for interactions between software elements is a way that facilitates debugging complex systems as once suggested by Hines ([0030]).

6. **As to claim 27**, Logan does not disclose a programming tool for creating and providing a graphic representation in a diagram of programs that control the flow of a process, comprising: a coordination element that manages interactions of objects that are involved in the control of the process and manages a sequence of the object interactions over time; and a display device that provides a graphic representation of the object interactions together with a graphic representation of the sequence of the object interactions over time in the diagram.

However, in an analogous art, Hines discloses a programming tool for creating and providing a graphic representation in a diagram of programs that control the flow of a process, comprising: a coordination element that manages interactions of objects that are involved in the control of the process ([0125], Lines 1-2; [0135], Lines 1-8; [0176], Lines 1-2, 5-13; [0221], Lines 1-6) and manages a sequence of the object interactions over time (Fig. 25, elements 2502, 2504; [0401], Lines 1-4); and a display device that provides a graphic representation of the object interactions together with a graphic representation of the sequence of the object interactions over time in the diagram (Fig. 33; Fig. 35; [0440] through [0444]).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Logan and the teachings of Hines to further provide a programming tool for creating and providing a graphic representation in a diagram of programs that control the flow of a process, comprising: a coordination element that manages interactions of objects that are involved in the control of the process and manages a sequence of the object interactions over time; and a display device that provides a graphic representation of the object interactions together with a graphic representation of the sequence of the object interactions over time in the diagram in Logan system.

The motivation is the need remains for a design and programming methodology that allows for coordination between software elements without tailoring the software element to the specific coordination style used in a particular software system while allowing for interactions between software elements is a way that facilitates debugging complex systems as once suggested by Hines ([0030]).

7. **As to claims 2, 21, and 28**, Logan discloses a programming tool wherein the process is an automation technology process ([0007]; [0033], Lines 1-7).

8. **As to claims 3, 22, and 29**, Logan discloses a method of programming and a programming tool wherein the process is a technical process ([0007]; [0033], Lines 1-7).

9. **As to claim 4**, Logan discloses a programming tool wherein the program is executed in plural, distributed stored program controllers (Fig. 1, elements of 18, 20, 22, ROBOT #1, ROBOT #2, ROBOT #3; [0009]; [0018]).

10. **As to claim 30**, Logan discloses a programming tool further comprising plural distributed stored program controllers in which the program is executed (Fig. 1, elements of 10, 18, 20, 22; [0009]; [0018]).

11. **As to claims 5 and 31**, Logan does not disclose a programming tool wherein a virtual or additional real processor is provided as the coordination element.

However, in an analogous art, Hines discloses a programming tool wherein a virtual or additional real processor is provided as the coordination element (Fig. 4B; [0133]; [0123]; ([0125], Lines 1-2; [0135], Lines 1-8; [0176], Lines 1-2, 5-13; [0221], Lines 1-6).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Logan and the teachings of Hines to further provide a programming tool wherein a virtual or additional real processor is provided as the coordination element in Logan system.

The motivation is the need remains for a design and programming methodology that allows for coordination between software elements without tailoring the software element to the specific coordination style used in a particular software system while allowing for interactions between software elements is a way that facilitates debugging complex systems as once suggested by Hines ([0030]).

12. **As to claims 6 and 32**, Logan does not disclose a programming tool wherein a virtual processor or an additional real processor is provided as the coordination element in connection with the distributed stored program controllers.

However, in an analogous art, Hines discloses a programming tool wherein a virtual processor or an additional real processor is provided as the coordination element in connection with the distributed stored program controllers (Fig. 4B; [0003] – distributed and embedded system methodologies; [0133]; [0123]; ([0125], Lines 1-2; [0135], Lines 1-8; [0176], Lines 1-2, 5-13; [0221], Lines 1-6).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Logan and the teachings of Hines to further provide a programming tool wherein a virtual processor or an additional real processor is provided as the coordination element in connection with the distributed stored program controllers.

The motivation is the need remains for a design and programming methodology that allows for coordination between software elements without tailoring the software element to the specific coordination style used in a particular software system while allowing for interactions between software elements is a way that facilitates debugging complex systems as once suggested by Hines ([0030]).

13. **As to claim 7 and 33**, Logan does not disclose a programming tool wherein at least substantially all calls of the objects are processed by the coordination element.

However, in an analogous art, Hines discloses a programming tool wherein at least substantially all calls of the objects are processed by the coordination element ([0176]; [0177]; [0221], Lines 1-6; [0229]; [0316]; Fig. 10; Fig. 14; [0317]).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Logan and the teachings of Hines to further provide a programming tool wherein at least substantially all calls of the objects are processed by the coordination element in Logan system.

The motivation is the need remains for a design and programming methodology that allows for coordination between software elements without tailoring the software element to the specific coordination style used in a particular software system while allowing for interactions between software elements is a way that facilitates debugging complex systems as once suggested by Hines ([0030]).

14. **As to claims 8 and 34**, Logan does not disclose a programming tool wherein the coordination element determines at least one of the instant of each call and the addressee of each call.

However, in an analogous art, Hines discloses a programming tool wherein the coordination element determines at least one of the instant of each call and the addressee of each call ([0018], Lines 1-3; [0031], Lines 1-5; Fig. 2; [0122], Lines 11-20)

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Logan and the teachings of Hines to further provide a programming tool wherein the coordination element determines at least one of the instant of each call and the addressee of each call in Logan system.

The motivation is the need remains for a design and programming methodology that allows for coordination between software elements without tailoring the software element to the specific coordination style used in a particular software system while allowing for interactions between software elements is a way that facilitates debugging complex systems as once suggested by Hines ([0030]).

15. **As to claim 9**, Logan discloses a programming tool wherein the graphics language comprises a graphic representation (Fig. 6B; Figs. 7-8) of all of the objects and a graphic representation of all of the object interactions, wherein each graphic representation, of the objects and the object interactions, respectively, is called and interconnected using an editor (Fig. 3, element 94; Fig. 5, element 142; Fig. 6B; Figs. 7-8; [0033], Lines 7-17) to implement an executable program (Fig. 3, elements 60, 98; Fig. 5, element 144).

16. **As to claim 35**, Logan discloses a programming tool further comprising an editor (Fig. 3, element 94; Fig. 5, element 142; Fig. 6B; Figs. 7-8; [0033], Lines 7-17) that calls and interconnects the graphic representation of the objects and the graphic

representation of the object interactions, respectively, to implement an executable program (Fig. 3, elements 60, 98; Fig. 5, element 144).

17. **As to claims 10 and 36**, Logan discloses a programming tool wherein each graphic representation in the diagram of an object and an object interaction is associated with an instruction or a program module ([0003], Lines 1-9; [0021], Lines 1-7; [0026] – there is a correspondence between the displayed flowchart, the graphic memory, the editor memory and the object memory in the Executive program).

18. **As to claims 11 and 37**, Logan discloses a programming tool wherein the instruction or the program module is in machine language ([0019], Lines 1-7).

19. **As to claims 12 and 38**, Logan discloses loop or jump to repeat at least one of an instruction ([0004]; Fig. 2A, element 58) and a program segment (Fig. 4A, element 104; Fig. 5, element 144); are each represented conditionally or unconditionally in the diagram (Fig. 2A, element 52; [0022], Lines 1-9) and are thereby implemented correspondingly ([0001]).

But Logan does not disclose a programming tool wherein the following additional object interactions: branching of an object call; parallel connection of an object call; synchronized connection of at least two interactions.

However, in an analogous art, Hines discloses a programming tool wherein the following additional object interactions: branching of an object call (Fig. 65A; [0114]; Fig.

66A; [0116]; [0601], Lines 3-10); parallel connection of an object call ([0018]; Fig. 21A; [0370]); synchronized connection (Fig. 8; [0046]; [0256] through [0259]) of at least two interactions.

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Logan and the teachings of Hines to further provide a programming tool wherein the following additional object interactions: branching of an object call; parallel connection of an object call; synchronized connection of at least two interactions.

The motivation is the need remains for a design and programming methodology that allows for coordination between software elements without tailoring the software element to the specific coordination style used in a particular software system while allowing for interactions between software elements is a way that facilitates debugging complex systems as once suggested by Hines ([0030]).

20. **As to claims 13 and 39**, Logan does not disclose a programming tool wherein a representation of the graphics language in the diagram shows the object interactions a first axis, and shows a sequence of the object interactions over time on a second axis of the diagram.

However, in an analogous art, Hines discloses a programming tool wherein a representation of the graphics language in the diagram shows the object interactions a first axis (Fig. 25, element 2502), and shows a sequence of the object interactions (Fig.

25, element 2512) over time on a second axis (Fig. 25, element 2504) of the diagram ([0401]).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Logan and the teachings of Hines to further provide a programming tool wherein a representation of the graphics language in the diagram shows the object interactions a first axis, and shows a sequence of the object interactions over time on a second axis of the diagram in Logan system.

The motivation is the need remains for a design and programming methodology that allows for coordination between software elements without tailoring the software element to the specific coordination style used in a particular software system while allowing for interactions between software elements is a way that facilitates debugging complex systems as once suggested by Hines ([0030]).

21. **As to claim 14**, Logan discloses a programming tool wherein the representation of the graphics language in the diagram is real-time capable (Fig. 1, element 10; [0041]).

22. **As to claims 18, 41, and 42**, Logan discloses a programming tool wherein the graphics language in the diagram can be constructed in real time (Fig. 1, element 10; [0041]).

23. **As to claims 15 and 43**, Logan discloses a programming tool wherein the display device is associated with a buffer memory (Fig. 3, element 66; Fig. 5, element 138; [0003]; [0018]; [0026]; [0028]) for buffered representation of the flow of the process using the graphics language.

24. **As to claims 16 and 44**, Logan does not disclose a programming tool wherein a sequence chart representation is selected as the diagram.

However, in an analogous art, Hines discloses a programming tool wherein a sequence chart representation is selected as the diagram (Figs. 25-30; [0401]).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Logan and the teachings of Hines to further provide a programming tool wherein a sequence chart representation is selected as the diagram in Logan system.

The motivation is the need remains for a design and programming methodology that allows for coordination between software elements without tailoring the software element to the specific coordination style used in a particular software system while allowing for interactions between software elements is a way that facilitates debugging complex systems as once suggested by Hines ([0030]).

25. **As to claims 17 and 40**, Logan does not disclose a programming tool wherein the diagram shows the sequence of object interactions over time on the second axis of the diagram from top to bottom.

However, in an analogous art, Hines discloses a programming tool wherein the diagram shows the sequence of object interactions (Fig. 25, elements 2512, messages) over time on the second axis (Fig. 25, element 2504) of the diagram from top to bottom.

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Logan and the teachings of Hines to further provide a programming tool wherein the diagram shows the sequence of object interactions over time on the second axis of the diagram from top to bottom in Logan system.

The motivation is the need remains for a design and programming methodology that allows for coordination between software elements without tailoring the software element to the specific coordination style used in a particular software system while allowing for interactions between software elements is a way that facilitates debugging complex systems as once suggested by Hines ([0030]).

26. **As to claim 20**, Logan does not disclose a method wherein the objects and the object interactions are arranged on a first axis of the common diagram, and wherein the successive sequence of the object interactions over time is represented by arranging the object interactions on a second axis of the common diagram.

However, in an analogous art, Hines discloses a method wherein the objects and the object interactions are arranged on a first axis (Fig. 25, element 2502) of the common diagram, and wherein the successive sequence of the object interactions (Fig.

25, elements 2512, messages; Figs. 26-30) over time is represented by arranging the object interactions on a second axis (Fig. 25, element 2504) of the common diagram.

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Logan and the teachings of Hines to further provide a method wherein the objects and the object interactions are arranged on a first axis of the common diagram, and wherein the successive sequence of the object interactions over time is represented by arranging the object interactions on a second axis of the common diagram in Logan system.

The motivation is the need remains for a design and programming methodology that allows for coordination between software elements without tailoring the software element to the specific coordination style used in a particular software system while allowing for interactions between software elements is a way that facilitates debugging complex systems as once suggested by Hines ([0030]).

27. **As to claim 23**, Logan does not disclose a method for programming wherein at least one of the arrangement of the objects and the object interactions, and the representation of the successive sequence of the object interactions over time, in the common diagram are real-time capable.

However, in an analogous art, Hines discloses a method for programming wherein at least one of the arrangement of the objects and the object interactions, and the representation of the successive sequence of the object interactions over time (Fig.

25; [0401]; Figs. 26-30), in the common diagram are real-time capable (Fig. 27; [0404]; Fig. 28; [0405]).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Logan and the teachings of Hines to further provide a method for programming wherein at least one of the arrangement of the objects and the object interactions, and the representation of the successive sequence of the object interactions over time, in the common diagram are real-time capable in Logan system.

The motivation is the need remains for a design and programming methodology that allows for coordination between software elements without tailoring the software element to the specific coordination style used in a particular software system while allowing for interactions between software elements is a way that facilitates debugging complex systems as once suggested by Hines ([0030]).

28. **As to claim 24**, Logan does not disclose a method for programming wherein a sequence chart representation is selected as the common diagram.

However, in an analogous art, Hines discloses a method for programming wherein a sequence chart representation (Fig. 25; [0401]; Figs. 26-30) is selected as the common diagram.

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Logan and the teachings of

Hines to further provide a method for programming wherein a sequence chart representation is selected as the common diagram in Logan system.

The motivation is the need remains for a design and programming methodology that allows for coordination between software elements without tailoring the software element to the specific coordination style used in a particular software system while allowing for interactions between software elements is a way that facilitates debugging complex systems as once suggested by Hines ([0030]).

29. **As to claim 25**, Logan does not disclose a method for programming wherein the common diagram is two-dimensional.

However, in an analogous art, Hines discloses a method for programming wherein the common diagram is two-dimensional (Fig. 25, elements 2502, 2504; [0401]).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Logan and the teachings of Hines to further provide a method for programming wherein the common diagram is two-dimensional in Logan system.

The motivation is the need remains for a design and programming methodology that allows for coordination between software elements without tailoring the software element to the specific coordination style used in a particular software system while allowing for interactions between software elements is a way that facilitates debugging complex systems as once suggested by Hines ([0030]).

30. **As to claim 26**, Logan does not disclose a method for programming wherein the successive sequence of the object interactions over time is represented by arranging the object interactions from top to bottom on the second axis of the common diagram.

However, in an analogous art, Hines discloses a method for programming wherein the successive sequence of the object interactions over time (Fig. 25; [0401]) is represented by arranging the object interactions from top to bottom (Fig. 25, elements 2512, messages; Figs. 26-30) on the second axis (Fig. 25, element 2504) of the common diagram.

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Logan and the teachings of Hines to further provide a method for programming wherein the successive sequence of the object interactions over time is represented by arranging the object interactions from top to bottom on the second axis of the common diagram in Logan system.

The motivation is the need remains for a design and programming methodology that allows for coordination between software elements without tailoring the software element to the specific coordination style used in a particular software system while allowing for interactions between software elements is a way that facilitates debugging complex systems as once suggested by Hines ([0030]).

Conclusion

31. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

- Chandhoke et al., *System and Method for Programmatically Generating a graphical Program Based on a Sequence of Motion Control, Machine Vision, and Data Acquisition (DAQ) Operations* (Pub. No. US 2006/0150149 A1)
- Rybarczyk et al., *Graphical Tool for Creating Discrete Phase Sequences and Device Control* (Pat. No. 6,041,178)

32. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Ben C. Wang whose telephone number is 571-270-1240. The examiner can normally be reached on Monday - Friday, 8:00 a.m. - 5:00 p.m., EST.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q. Dam can be reached on 571-272-3695. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

[illegible]
TUAN DAM
SUPERVISORY PATENT EXAMINER

BCW [illegible]

January 29, 2007